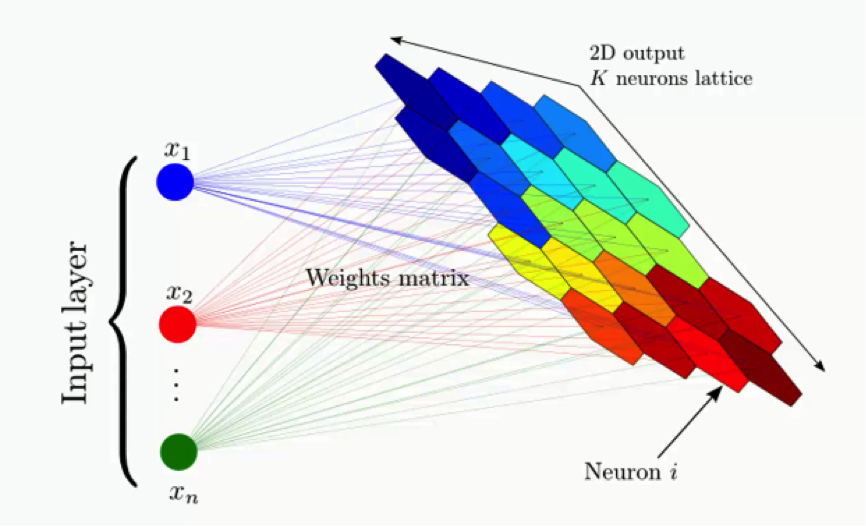
**Project: Anomaly detection using SOM and ANN**

**HOW DO SELF-ORGANIZING MAPS (SOM) WORK?**

The short answer would be **reducing dimensionality**.

The example below of a SOM comes from a paper discussing an amazingly interesting application of self-organizing maps in astronomy.

**[](http://www.superdatascience.com/wp-content/uploads/2018/09/Deep-Learning-A-Z%E2%84%A2-Self-Organizing-Maps-SOMs-How-do-Self-Organizing-Maps-Work_Img3.png)**

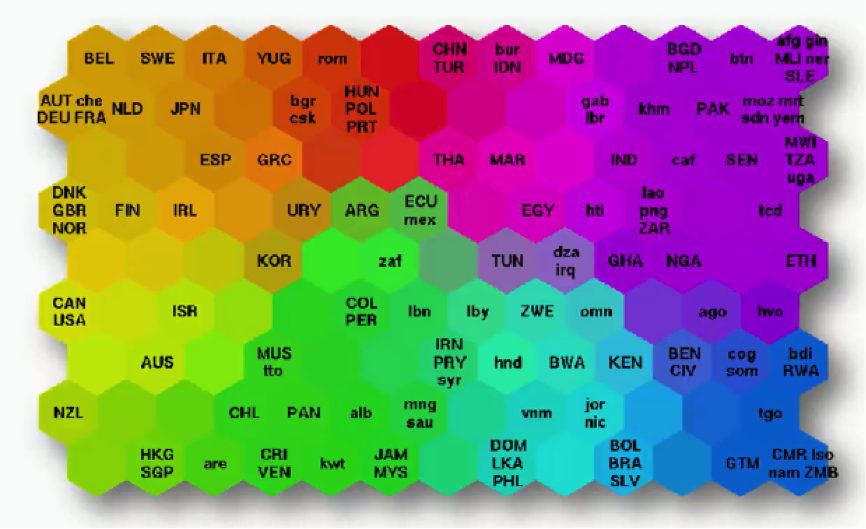
The example shows a complex data set consisting of a massive amount of columns and dimensions and demonstrates how that data set's dimensionality can be reduced.

So, instead of having to deal with hundreds of rows and columns (because who would want that!), the data is processed into a simplified map; that's what we call a self-organizing map.

The map provides you with a two-dimensional representation of the exact same data set; one that is easier to read.

**What exactly is the outcome to be gained from this process?**

What you see below is an actual SOM. This map represents the levels of economic wellbeing across a wide range of countries.

**[](http://www.superdatascience.com/wp-content/uploads/2018/09/Deep-Learning-A-Z%E2%84%A2-Self-Organizing-Maps-SOMs-How-do-Self-Organizing-Maps-Work_Img4.png)**

As you can see, the countries are lined up in a cluster, and their order inside that cluster is based on various indicators (e.g. health conditions, quality of education, average income per capita, etc.).

You can imagine this data set of 200+ countries to have started off with 39 different columns.

It's tough to imagine such a bulk of data, isn't it?

***That's exactly why we use SOMs.***

Instead of presenting you with each indicator separately, the SOM lines up the countries inside this cluster to make up a spectrum of wellbeing.

You can see the countries on the left side being the ones with the best economic statuses, although they vary in colors based on their positions within the category of high-income countries.

Then as you move to the right, you find the countries getting poorer until you reach the far right which contains the countries with the direst levels of poverty.

There's an important difference between SOMs as an unsupervised deep learning tool and the supervised deep learning methods like convolutional neural networks that you need to understand.

While in supervised methods the network is given a labeled data set that helps it classify the data into these categories, SOMs receive unlabeled data.

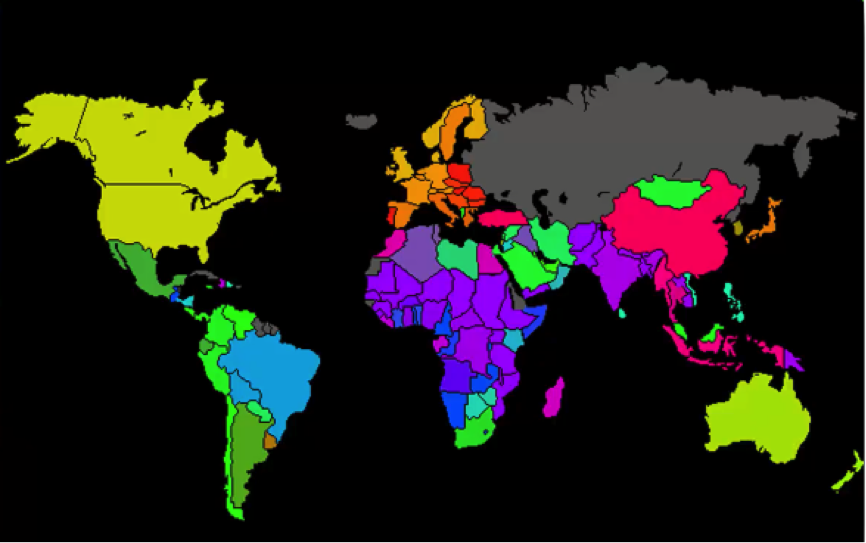
The SOM is given the data set and it has to learn how to classify its components on its own.

In this example, it would look at, say, Norway, Sweden, and Belgium, and left to its own devices, it learns that these countries belong together within the same area on the spectrum.

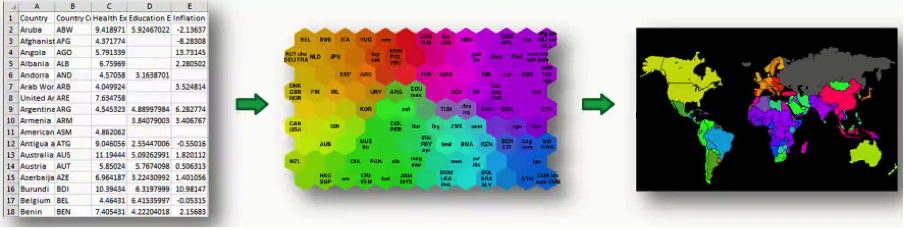
Besides giving us an easy way to read this data, SOMs have a wider range of uses.

For instance, you probably saw a map similar to the one below where countries are colored based on their human development scores, crime levels, military spending, etc.

Such maps can be colored using a SOM, since we can take the colors that the SOM chose for each group and apply it to our world map.

**[](http://www.superdatascience.com/wp-content/uploads/2018/09/Deep-Learning-A-Z%E2%84%A2-Self-Organizing-Maps-SOMs-How-do-Self-Organizing-Maps-Work_Img5.png)**

***Here's how the process goes:***

**[](http://www.superdatascience.com/wp-content/uploads/2018/09/Deep-Learning-A-Z%E2%84%A2-Self-Organizing-Maps-SOMs-How-do-Self-Organizing-Maps-Work_Img6.png)**

In this case, we got the data set from the World Bank containing all 39 indicators of human development, processed them into a SOM, and then applied that to a world map.

*As simple as that!*

If you want to study this process at a more detailed level, we would recommend a [**paper**](http://sci2s.ugr.es/keel/pdf/algorithm/articulo/1990-Kohonen-PIEEE.pdf) titled “The Self-Organizing Map” that was authored by Tuevo Koheonen himself back in 1990, and a slightly deeper introduction to SOMs, you can read this [**post**](http://www.ai-junkie.com/ann/som/som1.html) from 2004 by Mat Buckland.

**Data set:**

<http://archive.ics.uci.edu/ml/datasets/statlog+(australian+credit+approval)>

**The code:**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Credit\_Card\_Applications.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

# Feature Scaling

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature\_range = (0, 1))

X = sc.fit\_transform(X)

# Training the SOM

from minisom import MiniSom

som = MiniSom(x = 10, y = 10, input\_len = 15, sigma = 1.0, learning\_rate = 0.5)

som.random\_weights\_init(X)

som.train\_random(data = X, num\_iteration = 100)

# Visualizing the results

from pylab import bone, pcolor, colorbar, plot, show

bone()

pcolor(som.distance\_map().T)

colorbar()

markers = ['o', 's']

colors = ['r', 'g']

for i, x in enumerate(X):

w = som.winner(x)

plot(w[0] + 0.5,

w[1] + 0.5,

markers[y[i]],

markeredgecolor = colors[y[i]],

markerfacecolor = 'None',

markersize = 10,

markeredgewidth = 2)

show()

# Finding the frauds

mappings = som.win\_map(X)

frauds = np.concatenate((mappings[(2,5)], mappings[(3,6)]), axis = 0)

print frauds

print ''''''''

frauds = sc.inverse\_transform(frauds)

print frauds

# Part 2 - Going from Unsupervised to Supervised Deep Learning

# Creating the matrix of features

customers = dataset.iloc[:, 1:].values

# Creating the dependent variable

is\_fraud = np.zeros(len(dataset))

for i in range(len(dataset)):

if dataset.iloc[i,0] in frauds:

is\_fraud[i] = 1

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

customers = sc.fit\_transform(customers)

# Part 2 - Now let's make the ANN!

# Importing the Keras libraries and packages

from keras.models import Sequential

from keras.layers import Dense

# Initialising the ANN

classifier = Sequential()

# Adding the input layer and the first hidden layer

classifier.add(Dense(units = 2, kernel\_initializer = 'uniform', activation = 'relu', input\_dim = 15))

# Adding the output layer

classifier.add(Dense(units = 1, kernel\_initializer = 'uniform', activation = 'sigmoid'))

# Compiling the ANN

classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

# Fitting the ANN to the Training set

classifier.fit(customers, is\_fraud, batch\_size = 1, epochs = 2)

# Predicting the probabilities of frauds

y\_pred = classifier.predict(customers)

y\_pred = np.concatenate((dataset.iloc[:, 0:1].values, y\_pred), axis = 1)

y\_pred = y\_pred[y\_pred[:, 1].argsort()]

print y\_pred